IN THE CLAIMS

Please amend the claims as indicated below:

1. (currently amended) A method of optimizing compiled code generated from high level computer programming languages, wherein the compiled code includes loop constructs, the method comprising the steps:

(1) providing a non-optimized loop code segment corresponding to a loop construct written in a high level programming language, wherein in the non-optimized loop code segment the loop construct is executed a loop repetition number of times *n* and the non-optimized loop code segment includes a call to a procedure~~, the call depending on a number of arguments, wherein the call invokes the procedure only if a certain condition is met and wherein the certain condition includes one of the arguments being less than another one of the arguments~~;

(2) providing execution conditions such that program variables of the compiled code have certain values satisfying initial and terminating conditions for ~~required to~~ causinge execution of the loop construct the loop repetition number of times *n*, wherein the call invokes the procedure responsive to arguments of the call, the arguments including ones of the program variables;

(3) optimizing the non-optimized loop code segment for the execution conditions to provide a consolidated code segment corresponding with the execution conditions for execution of the loop said loop repetition number of times *n,* wherein the consolidated code includes certain code of the non-optimized loop code segment and omits certain other code of the non-optimized loop code segment and wherein the call is omitted from the consolidated loop code segment if the certain program variable values are such that the call invoking the procedure results in no change in program variables ~~execution conditions indicate the certain condition is not met~~;

(4) determining whether the consolidated code segment should be executed in preference to the non-optimized loop code segment ; and

(5) if said determination is favourable, including the consolidated code segment in optimized code for a program written in the high level programming language.

2. (currently amended) A method of optimizing the compiled code generated from high level computer programming languages, wherein the compiled code includes loop constructs, the method comprising the steps:

(1) providing a non-optimized loop code segment corresponding to a loop construct written in a high level programming language, wherein in the non-optimized loop code segment the loop construct is executed a loop repetition number of times *n* and the non-optimized loop code segment includes a call to a procedure~~, the call depending on a number of arguments, wherein the call invokes the procedure only if a certain condition is met and wherein the certain condition includes one of the arguments being less than another one of the arguments~~;

(2) providing a non-optimized pre-loop code segment corresponding to programming instructions preceding the loop construct, and a non-optimized post-loop code segment corresponding to instructions succeeding the loop construct;

(3) providing execution conditions such that program variables of the compiled code have certain values satisfying initial and terminating conditions for ~~required to~~ causing~~e~~ execution of the loop construct the loop repetition number of times *n*, wherein the call invokes the procedure responsive to arguments of the call, the arguments including ones of the program variables;

(4) revising the non-optimized pre-loop, loop and post-loop code segments to include the execution conditions; and

(5) optimizing the non-optimized pre-loop, loop and post-loop code segments for the execution conditions to provide a consolidated code segment corresponding with the execution conditions for execution of the loop said loop repetition number of times *n*, wherein the consolidated code includes certain code of the non-optimized loop code segment and omits certain other code of the non-optimized loop code segment and wherein the call is omitted from the consolidated loop code segment if the certain program variable values are such that the call invoking the procedure results in no change in program variables~~execution conditions indicate the certain condition is not met~~;

(6) determining whether the consolidated code segment should be executed in preference to the non-optimized code segments; and

(7) if said determination is favourable, including the consolidated code segment in optimized code for a program written in the high level programming language.

3. (previously presented) The method as claimed in claim 1, wherein said determination involves a cost-benefit analysis to determine whether the cost of using the consolidated code segment is reduced by a predetermined threshold compared with not using the consolidated code segment.

4. (original) The method as claimed in claim 1, wherein the inclusion of said consolidated code segment in the optimized code is conditional on the occurrence of the execution conditions.

5. (original) The method as claimed in claim 1, wherein said loop constructs includes any one or more of the following loop constructs: for loops, while loops, repeat loops.

6. (original) The method of claim 1, wherein said steps (1) to (5) are repeated a predetermined number of times *k*, for values of the loop repetition number *n* from 0 to *k*-1.

7. (previously presented) The method as claimed in claim 2, wherein said determination involves a cost-benefit analysis to determined whether the cost of using the consolidated code segment is reduced by a predetermined threshold compared with not using the consolidated code segment.

8. (original) The method as claimed in claim 2, wherein the inclusion of said consolidated code segment in the optimized code is conditional on the occurrence of the execution conditions.

9. (original) The method as claimed in claim 2, wherein said loop constructs includes any one or more of the following loop constructs: for loops, while loops, repeat loops.

10. (original) The method of claim 2, wherein said steps (1) to (7) are repeated a predetermined number of times *k*, for values of the loop repetition number *n* from 0 to *k*-1.

11. (currently amended) A compiler for optimizing the compiled code generated from high level computer programming languages, wherein the compiled code includes loop constructs, the compiler being embodied on a computer-readable medium, the compiler comprising:

(1) compiler code means for providing a non-optimized loop code segment corresponding to a loop construct written in a high level programming language, wherein in the non-optimized loop code segment the loop construct is executed a loop repetition number of times *n* and the non-optimized loop code segment includes a call to a procedure~~, the call depending on a number of arguments, wherein the call invokes the procedure only if a certain condition is met and wherein the certain condition includes one of the arguments being less than another one of the arguments~~;

(2) compiler code means for providing execution conditions such that program variables of the compiled code have certain values satisfying initial and terminating conditions for ~~required to~~ causing~~e~~ execution of the loop construct the loop repetition number of times *n*, wherein the call invokes the procedure responsive to arguments of the call, the arguments including ones of the program variables;

(3) compiler code means for optimizing the non-optimized loop code segment for the execution conditions to provide a consolidated code segment corresponding with the execution conditions for execution of the loop said loop repetition number of times *n*, wherein the consolidated code includes certain code of the non-optimized loop code segment and omits certain other code of the non-optimized loop code segment and wherein the call is omitted from the consolidated loop code segment if the certain program variable values are such that the call invoking the procedure results in no change in program variables~~execution conditions indicate the certain condition is not met~~;

(4) compiler code means for determining whether the consolidated code segment should be executed in preference to non-optimized code segments; and

(5) compiler code means for including the consolidated code segment in optimized code for a program written in the high level programming language, if said determination is favourable.

12. (currently amended) A compiler for optimizing the compiled code generated from high level computer programming languages wherein the compiled code includes loop constructs, the compiler being embodied on a computer-readable medium, the compiler comprising:

1) compiler code means for providing a non-optimized loop code segment corresponding to a loop construct written in a high level programming language, wherein in the non-optimized loop code segment the loop construct is executed a loop repetition number of times *n* and the non-optimized loop code segment includes a call to a procedure~~, the call depending on a number of arguments, wherein the call invokes the procedure only if a certain condition is met and wherein the certain condition includes one of the arguments being less than another one of the arguments~~;

2) compiler code means for providing a non-optimized pre-loop code segment corresponding with programming instructions preceding the loop construct, and a non-optimized post-loop code segment corresponding with instructions succeeding the loop construct;

3) compiler code means for providing execution conditions such that program variables of the compiled code have certain values satisfying initial and terminating conditions for ~~required to~~ causing~~e~~ execution of the loop construct the loop repetition number of times *n*, wherein the call invokes the procedure responsive to arguments of the call, the arguments including ones of the program variables;

4) compiler code means for revising the non-optimized pre-loop, loop and post-loop code segments to include the execution conditions; and

5) compiler code means for optimizing the non-optimized pre-loop, loop and post-loop code segments for the execution conditions to provide a consolidated code segment corresponding with the execution conditions for execution of the loop said loop repetition number of times *n*, wherein the consolidated code includes certain code of the non-optimized loop code segment and omits certain other code of the non-optimized loop code segment and

wherein the call is omitted from the consolidated loop code segment if the certain program variable values are such that the call invoking the procedure results in no change in program variables~~execution conditions indicate the certain condition is not met~~;

6) compiler code means for determining whether the consolidated code segment should be executed in preference to the non-optimized code segments; and

7) compiler code means for including the consolidated code segment in optimized code for a program written in the high level programming language, if said determination is favourable.

13. (previously presented) The compiler as claimed in claim 11, wherein said determination involves a cost-benefit analysis to determined whether the cost of using the consolidated code segment is reduced by a predetermined threshold compared with not using the consolidated code segment.

14. (original) The compiler as claimed in claim 11, wherein the inclusion of said consolidated code segment in the optimized code is conditional on the occurrence of the execution conditions.

15. (original) The compiler as claimed in claim 11, wherein said loop constructs includes any one or more of the following loop constructs: for loops, while loops, repeat loops.

16. (original) The compiler of claim 11, wherein said steps (1) to (5) are repeated a predetermined number of times *k*, for values of the loop repetition number *n* from 0 to *k*-1.

17. (previously presented) The compiler as claimed in claim 12, wherein said determination involves a cost-benefit analysis to determined whether the cost of using the consolidated code segment is reduced by a predetermined threshold compared with not using the consolidated code segment.

18. (original) The compiler as claimed in claim 12, wherein the inclusion of said consolidated code segment in the optimized code is conditional on the occurrence of the execution conditions.

19. (original) The compiler as claimed in claim 12, wherein said loop constructs includes any one or more of the following loop constructs: for loops, while loops, repeat loops.

20. (original) The compiler of claim 12, wherein said steps (1) to (7) are repeated a predetermined number of times *k*, for values of the loop repetition number *n* from 0 to *k*-1.